1974-09

A microprogrammed diskette control unit

Darmawan

Monterey, California. Naval Postgraduate School

A MICROPROGRAMMED DISKETTE CONTROL UNIT

Darmawan

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A Microprogrammed Diskette Control Unit

by

Darmawan

September 1974

Thesis Advisor: V. Michael Powers

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Microprogrammed Diskette Control Unit		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; September 1974 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Darmawan		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE September 1974 13. NUMBER OF PAGES 52
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Microprogrammed disk controller
Disk controller
Diskette
Floppy disk

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Considerations in the design of microprogrammed disk controllers are presented. First, the concept of microprogramming and the characteristics of diskette storage drives are discussed. The use of disk storage in data base management systems is also discussed. Finally, the functions, programs and structure of a microprogrammed disk controller for personnel file management are presented.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
(Page 1) S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

A Microprogrammed Diskette Control Unit

by

Darmawan
Major, Indonesian Navy

Submitted in partial fulfillment of the
requirements for the degree of

NASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the
NAVAL POSTGRADUATE SCHOOL
September 1974

ABSTRACT

Considerations in the design of microprogrammed disk controllers are presented. First, the concept of microprogramming and the characteristics of diskette storage drives are discussed. The use of disk storage in data base management systems is also discussed. Finally, the functions, programs and structure of a microprogrammed disk controller for personnel file management are presented.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

I. INTRODUCTION

The development of microprogramming and its various uses have received wide international interest. The advantages of systems which use microprogramming are more and more interesting. Its application is not limited to the control section of a computer system but also is appropriate for the control section of a peripheral device control unit. The diskette is a low cost device, with ease of handling small jobs as well as big jobs. Data can be written to or read from the diskette at high speed, which is very useful in record (file) processing. Data management systems with various kinds of file organisation and processing are presented to ease the decision of which system is the most suitable for a typical data base management operation. Maintenance of a personnel file requires a large amount of data shuffling operations.

Finally, a simplified design of a microprogrammed diskette controller together with example microprograms for implementation of the functions in personnel file management are presented as one example of a real application of the microprogrammed peripheral control unit.

II. MICROPROGRAMMING

A. HISTORY

The original concept of a programmed control stored in nondestructive read only memory was proposed in 1951 by M. V. Wilkes of Cambridge University Mathematical Laboratory in a speech prepared for the Manchester University Computer Conference and entitled, "The Best Way to Design an Automatic Calculating Machine."

"Wilkes and his colleagues were clearly concerned with the ad hoc and nonsystematic approach that was applied to design the computer control."(Ref. 1) They sought a means for rearranging the computer components into a systematic order, easy to implement and maintain. They were more interested in simplifying the design task than in any hardware saving that could be realized.

The execution of a computer instruction involves a sequence of transfers of information between registers in the processors; some of these transfers take place directly and some through an adder or other logical circuit. The execution of these individual steps can be implemented as a program instead of as hard wired connections. This program which actually performs only one machine language instruction is called a microprogram.

The term microprogramming had also been used prior to 1950 to describe a system in which individual bits in an instruction control direct certain gates in the processor. Work by van der Poel(Ref. 13) at the Lincoln Laboratory falls into this category. Such a scheme gives the programmer a larger repertoire of instruction than he would normally have. The Wilkes scheme was first implemented in the control system built at the University of Cambridge and known as EDSAC.

In 1955 Billing and Hopmann (Ref. 6) published a paper in which they discussed general principles of programming and their practical application. In 1956 and 1957 two papers by Glantz (Ref. 7) and Mercer (Ref. 8) were published in which the stored logic concept was presented as a means for altering the microprogram. In 1958 Dineen (Ref. 9) and 2 years later Kampe (Ref. 10) described a simple computer based on the Wilkes model.

The years 1961 to 1964 were the peak years of international interest in microprogramming, not only in the US and England, but also in Italy, Japan, Russia, Australia and France. There were two Italian papers: Gerace described a microprogrammed processor built in Pisa, which was based on the use of ferrite core read only memory; Graseelli was concerned with the problem of how to construct a stored logic computer without using a very fast read-write memory. The Japanese paper by Hagiwara (Ref. 12) described the use of a diode matrix with a diode in each intersection instead of only at selected intersections. Each diode was connected in series with a photo transistor and could be switched in or out of the circuit by illuminating or darkening the transistors. Light was allowed to fall, through a perforated card, on only those transistors that were required to be conducting. Thus the microprogram in use could be replaced by another one by changing the perforated card. The Russian paper by Emelyanov Yaroslawsky was a recapitulation of the principles of microprogramming and its possibilities. Australia came in with a description of CIRCUS, a microprogrammed computer with quite an elaborate instruction set and interrupt scheme. The French paper was on the evolution of the microprogramming concept. In February 1964 the issue of Datamation contained a number of articles describing specific stored logic systems designed to extend the computing capability of a small machine.

The first microprogrammed application by any major computer manufacturer was made by IBM. The microprogramming of the IBM 7950 was presented in several papers at the 1961 ACM Conference.

In 1964 the IBM System 360 series had microprogramming based on a read only memory. In the following year, a paper appeared describing the microprogrammed RCA Spectra 70 series model 45.

In 1964 a complete discussion by Tucker of the microprogramming technique adopted in the design of the IBM system 360 was published. The advantages of microprogramming may be summarized as:

1. It provides economical means whereby the smaller machines of a series can have a large instruction set compatible with those of the larger one.

2. Maintenance aids can be provided. For example, the read only memory can have a parity bit, and special diagnostic microroutines can be provided for the use of maintenance engineers.

3. Emulation is possible.

4. Flexibility exists to provide new features in the future.

B. CONCEPT OF MICROPROGRAMMING

Microprogramming was originally conceived and proposed as an orderly, alternative design procedure to the ad hoc procedure applied to conventional ad hoc hardware. It eventually became an alternative design and implementation tool for the control section, where hardware control is replaced by a stored logic section or a microprogram control section stored in a high speed nondestructive read only memory.

Basically, microprogrammed control consists of two parts: control store and control decode (Ref. 2). The hardware for both is centrally located, rather than distributed as in the case of conventional control. The control

store may be a read only memory (ROM) or it may be a writeable control store. Stored in this memory, either permanently or semipermanently, are the microprograms. For the most part, the microprograms are dedicated to the execution of the machine instructions; however, some microprograms perform other functions.

To illustrate the concept of microprogramming, it is best to look at the hardware that is to be controlled, and at how the logic can be controlled by a microprogram. Fig. II-B-1 is a hypothetical computer data flow.

As an example, consider the steps to add the content of a memory location to the content of the A Register - to execute the ADD instruction. Assume that the ADD instruction has been fetched from memory. The first step is to ask memory to fetch the address operand.

1. Operand address in MD is read out to the B_1 bus and
2. Stored in M Reg.; also
3. A command is sent to memory to read the content of the addressed location.

The second step is to bring the operand from the memory into the central processor unit.

4. Read the register content out to the B_2 bus.
5. Store from the B_1 bus into the B Register.

The third step is to add the two operands and deposit the result in the A Register.

6. Read out the content of the A Register to B_2 bus.
7. Read out the content of B Register to B_1 bus.
8. Send an add command to the adder.
9. Store the result from the C bus into the A Register.

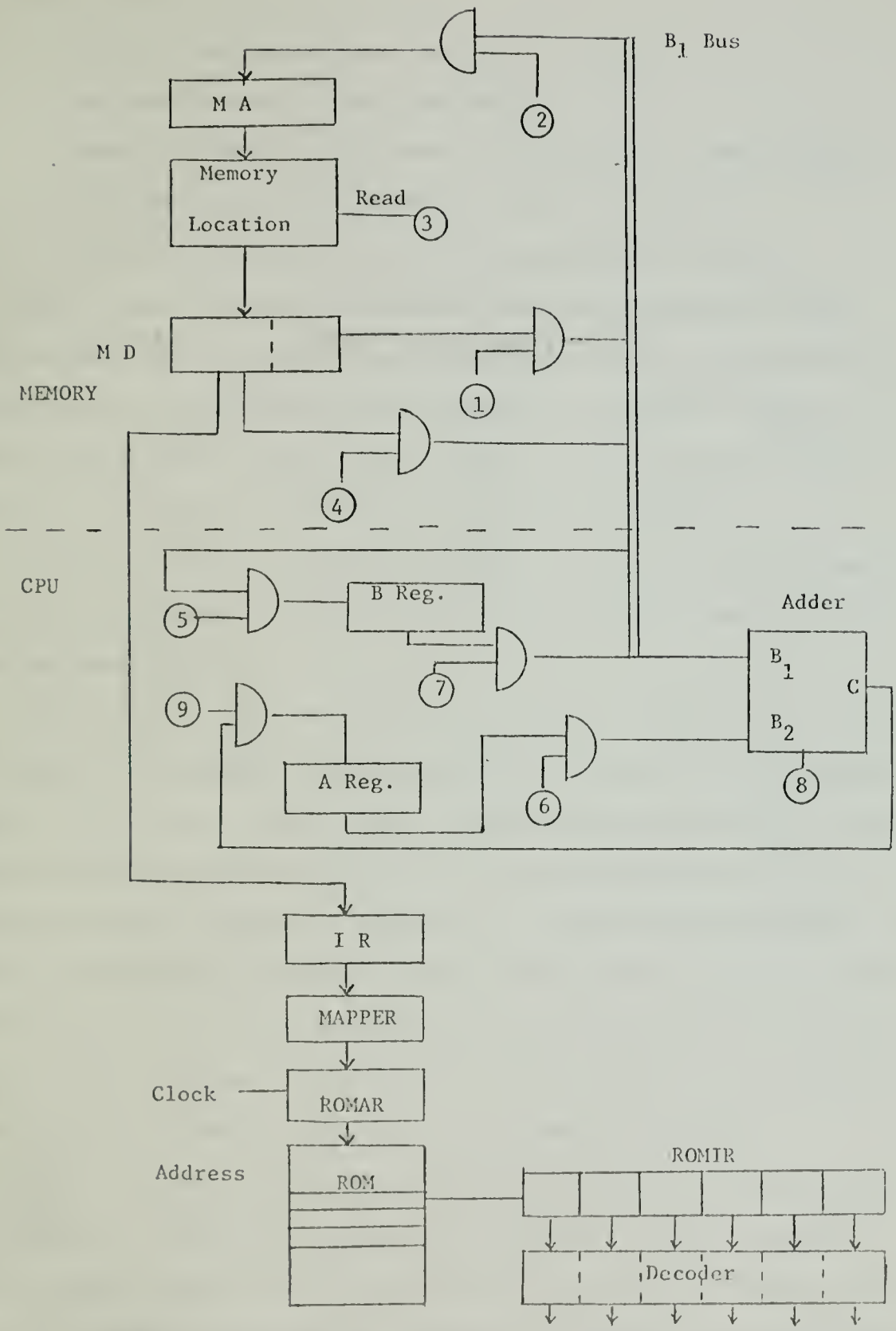


Fig. II-B-I. Hypothetical computer controls

These actions can be grouped into:

1. Read something onto B_1 bus (actions 1, 4 and 7).
2. Read something onto B_2 bus (action 6).
3. Command the adder to do a specific function (i.e., ADD) (action 8).
4. Store something into a register (actions 2, 5 and 9).
5. Cause special function (i.e., read from memory) (action 3).

Each of these is decoded from the ROM Instruction Register (ROMIR). Thus the ROMIR will be decoded into five separate fields. Combinations of several bits in the field permit the selection of a specific number of sources, destinations, or functions. There is one more important field which we do not use in the example; that is, Skip (SK) field.

The number of bits in each field depends on how many gates and functions are to be controlled. The number of gates also depends on how many instructions the computer has.

Since the B_1 bus and Store field are used three separate times, there will have to be a minimum of three microinstruction words. Thus the microprogram ADD consists of three storable microinstruction words which occupy location 0101, 0102, 0103 of the ROM. To execute the ADD instruction, it is only necessary to read the three words in succession into the ROMIR. This is done by applying the starting address of 0101 (which is derived from the ADD instruction code) to the ROM address Register (ROMAR) and then permitting the clock to increment the ROMAR three times. After this the ROMAR is normally forced to address 0000, rather than proceeding to 0104.

In one memory cycle, the ROM cycle can cause several microinstructions to be executed. Timing of the ROM is very important, since its operation is very closely related to the memory cycle of the computer program memory. Memory read usually starts at the beginning of the memory cycle, and memory

write in the middle. Thus, data will be available after the read cycle (when reading from memory) and must be loaded into MD at the end of read cycle. Memory instructions cannot always be executed in every consecutive ROM cycle because the microprogram must be delayed at certain points to synchronize with the memory or I/O cycle.

III. DATA MANAGEMENT SYSTEM

A. COMPUTER FILE

A record is composed of related data items, also termed fields. A collection of records is known as a file. A file is kept for a variety of purposes. Four main types are usually identified.

1. Master file, a relatively permanent record containing statistical identification, and historical information which is used as a source of reference. Examples of this type of record are: Accounts receivable file, personnel file, and inventory file.

2. Transaction file, also called detail file, is a collection of records from the processing transaction. Sales invoice file and purchases file are examples of this file.

3. A report file, a collection of records extracted from data in the master file in order to prepare a report. Example: Report file for taxes withheld, report file for analysis of employee skills.

4. A sort file, is a working file of records to be sequenced. This may be the original or a copy of a transaction file, a master file or a report file.

Each record in a file is identified by an identification field. This identification field, used as a basis for sequencing and searching the file, is frequently called the record key. This key can be numeric such as a social security number, or alphabetic, such as a name. There can be more than one key, and a record may therefore be sequenced as one key in one file and another key in a second file.

Some records may be divided into two parts: a master portion, containing relatively permanent data and a detailed trailer position. For example, an accounts receivable system which keeps track of individual invoices might have a master record portion containing fairly permanent customer data such as name, address, etc., plus several trailer records, each containing an unpaid invoice.

A record, as defined by record layout, may not be identical with the physical record on the file storage medium. The record layout defines a logical record, whereas storage is done in terms of physical records. A data item or field may vary in length, for example, a name. The storage requirement will vary because the names vary in length. This leads to two possibilities: fixed field size and variable field size. A variable length field is specified either by a length subfield at the beginning of the field or by a special symbol at the end.

The number of items in a record can also be fixed or variable. In the variable approach, there must be some method of identifying the items that are present. In the fixed number method all fields are present and if there is no data, the field is filled with spaces or some other fill character. The fixed item size and fixed number of items are easier to handle for programming and execution but require more storage.

B. FILE ORGANISATION AND PROCESSING

1. Sequential File Organisation

Sequential file organisation is the most common because it makes effective use of the least expensive file medium, like magnetic tape, and because sequential processing at periodic intervals using a batch of transactions is very efficient. In a sequential file the logical order of records in the file and the physical order of records on the recording medium are the same. Ordering can be in ascending or descending order by a key field which may be numeric or alphabetic. Since the records are physically ordered by the field, there is no storage location identification. The identification is by the key, not by the fact that the record is in the n-th on the storage medium.

The advantage of sequential organisation is fast access per relationship during retrieval. Since the record is ordered sequentially, a binary search is possible. The data is sampled in the middle, eliminating half the cases in one comparison; the remaining half is then sampled in its middle and the process is repeated until a sequential search of a small portion of the file is possible and it is established that the record is or is not present (Ref. 3)

The advantage becomes a disadvantage where a file is to be inserted or deleted. The insertion process requires the already stored record be pushed apart to make room for a new record, resulting in copying of the entire file. The converse is true for deleting records, in which case existing records are pushed together.

2. Random Organisation

In random data organisation, records are stored and retrieved on the basis of a predictable relationship between the key of the record and the direct address of the location where the record is stored. In direct access processing, a major problem is identifying the address where a record to be accessed is stored. The address can be obtained in several different ways.

1. Identification name as address.
2. Address supplied as separate input field.
3. Address derived by calculation on the record key.
4. Address located from an index.

Random organisation allows for rapid access to a record. However it requires extra processing time to obtain the address. If the records are of a uniform length, individual records can be stored, retrieved and updated without affecting other records on the storage medium.

3. List Organisation

In the sequential organisation, the next logical record is also the next physical records. In a list organisation pointers are used to connect logical records so that the logical organisation is divorced from the physical organisation on the medium. A pointer in the record consists of a data item which gives the address of another record. If the file is on a disk, the pointer to a record is the disk address of the record. There are three main types of list organisation: simple list, inverted list and ring. Fig. lII-B-1 is an example of a simple list which is stored in a logically sequential order. They are not, however, in the same physical order.

Records can be placed anywhere within a list by changing the pointer of the record preceding the new record and inserting a new pointer in the record just inserted. For example, if a record is to be inserted between the "DEF" and the "XYZ" record in Fig. III-B-1, the pointer in the DEF recorder would be set to the new record address and the new record pointer would be given the value 100, which points to the XYZ record. The deletion of a record from a list requires changing the pointer in the preceding record to point to the record following the one being deleted.

A fully inverted file is one in which all data items are in an index. Each data item in the index has a list of pointers to records which use that characteristic. The records themselves do not have pointers and do not need to have data fields because these are found in the index (see Fig. III-B-2). Since the fully inverted file makes every data item available as a key for information retrieval, it is excellent for data retrieval especially in cases where data retrieval requirements cannot be specified in advance. Usually fully inverted files will be excessively large. Easy retrieval, storing and updating data are more difficult because of the maintenance of a large index.

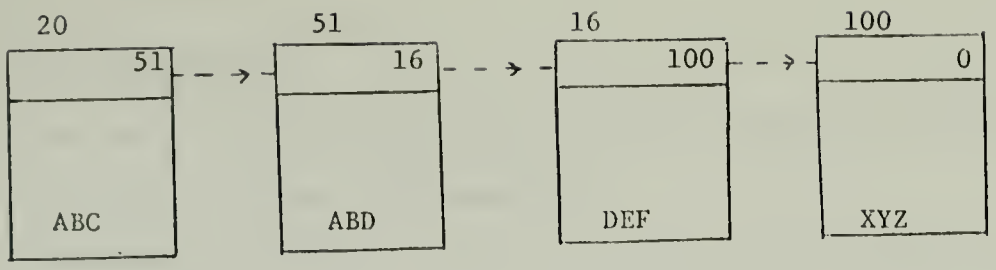


Fig. III-B-1. Simple List

INDEX

key	pointer
student	1,3,14
clerical	2,7,10
born '40	1,2
born '30	3,30
male	1,3,10
female	2,7
etc.	

1	Jones
2	Betty
3	Johnson
4	Henry
	etc.

Fig. III-B-2. Fully Inverted File

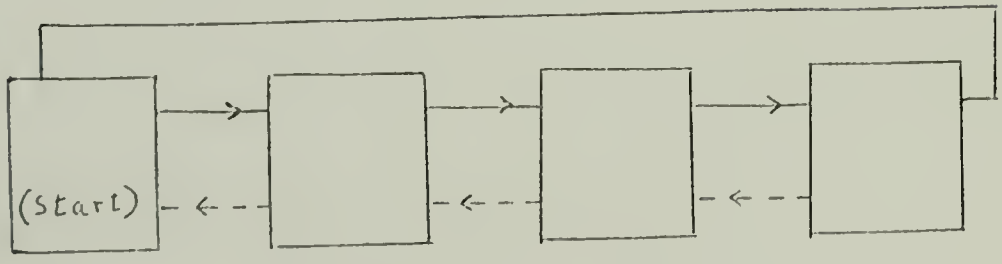


Fig. III-B-3. Ring Structure

C. FILE MANAGEMENT

The collection of all activities relating to the creation, updating and use of files is called file management.

Creation of the file is the activity of creating a new file or converting a file which already exists into a computer file medium. The conversion includes not only from a non-computer file, but also from one computer system to another.

Updating a file can be classified as file maintenance. Its activities include inserting a new record, changing existing records by adding, altering or deleting data, and deleting records.

Copying of selected records for analysis, report, etc., are examples of the uses of a file.

IV. DIRECT ACCESS STORAGE DEVICES

A. GENERAL

Direct access storage enables record retrieval without sequentially searching through a file. Thus, when only some of the records of a given run are active a direct access storage device provides significantly better throughput than a device that handles each record sequentially. Direct access storage also enables random inquiries during record processing. A direct access scheme can also process records sequentially. Therefore, with direct access storage devices, records can be stored for retrieval sequentially or randomly.

There are three types of direct access storage devices (DASD's): disk storage, drum storage and data cell storage. Data cell storage usually is used for batch, sequential, high volume activity. The drum storage provides large capacity storage and extremely fast access and the disk storage features the needs of general data processing applications.

All activities in the direct access storage devices are controlled by a storage control unit. Such a control unit can perform the following functions:

1. Interpret and execute commands from the channel attached to the central processing unit (CPU).

2. Provide a path for data between the standard computer interface and attached storage device.

3. Translate data appropriately as it is transferred between storage device and standard interface.

4. Furnish operational status information to the channel.

5. Check the accuracy of data transfer.

The disk storage device consists of drive mechanism, read and/or write head, track position mechanism and associated electronics. The medium for storing the data is the disk itself; it can be disk pack or a diskette.

The disk pack contains a number of disks; these disks are divided into cylinders that consist of a number of tracks. The beginning of a track is identified by an index mark, which is usually coincident with the leading edge of the physical index. A track can be divided into sectors. These sectors can be soft or hard sectors. Hard sector means that there are physical holes or notches or physical volume write protect. A soft sector does not have physical holes or notches or volume write protect; the sector is identified by an identification address mark. Soft sectoring allows greater flexibility with the use of variable length records. The track format is different for different disks and different controllers. Each field on a track is separated from adjacent fields by a number of bytes containing no data bits. These areas are referred to as gaps, which also provide time for control functions.

B. DISK CONTROL UNIT

1. Conventional Disk Control Unit

A block diagram of a conventional control unit (Ref. 4) is shown in Fig. IV-B-1. It consistsof channel interface, controller logic, data processing unit and device interface. The channel interface is a device which communicates with the channel and the device interface communicates with the direct access storage device. The data processing unit converts the signals from the device (DASD) into data usable by the channel and vice versa. The controller logic controls the activities of the data processing unit, channel interface and device interface.

Information is transferred between channel and control unit one byte at a time and one bit at a time between attached storage device and control unit. The basic functions are: seek, read and write. The seek command moves the access mechanism to the desired track (cylinder). The read command transfers information from the attached direct access storage device to the channel. The write command transfers data from CPU storage to a specified storage device. The data processing unit has three major components (Fig. IV-B-2)[Ref. 4].

1. General register (GR) which holds intermediate results.

2. Serializer/Deserializer (SERDES), which converts a byte of information from serial to parallel format or vice versa. SERDES also performs: index and address mark detection, clock pulses for writing, cyclic redundancy check (CRC) generation and detection.

3. Arithmatic logic unit (ALU) performs arithmetic and logic functions.

In seek, first obtain the content of track address register (TAR)[Fig. IV-B-3]; this content is then compared with the seek address (track to find) and the calculation is made to determine the number of tracks of travel required to reach the desired track. This quantity is sent to the track step counter (TSC) and the direction flip flop is set to indicate the direction of travel. Now the read/write head can travel to the desired track. Each time the head passes over a track, the content of the track step counter is decremented, and no further travel occurs if the content is zero.

Write begins when the R/W head is properly positioned. SERDES will convert parallel data into serial and modulate these data with the writing clock to give the proper recording format. Read is almost the same as write only the function of SERDES is reversed. All other functions are accomplished by using combinations of the three basic functions.

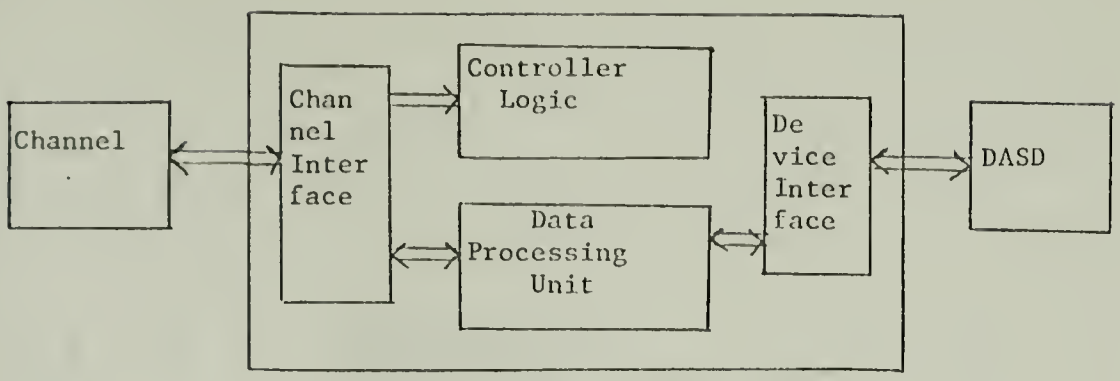


Fig. IV-B-1. Control Unit

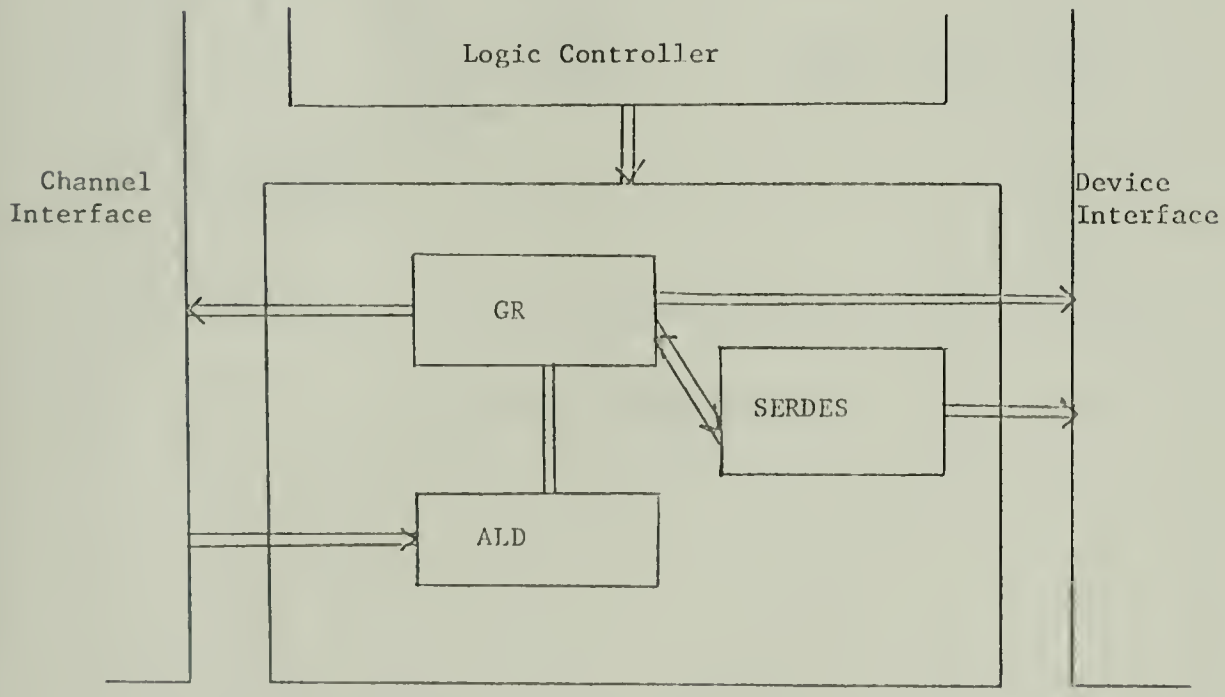
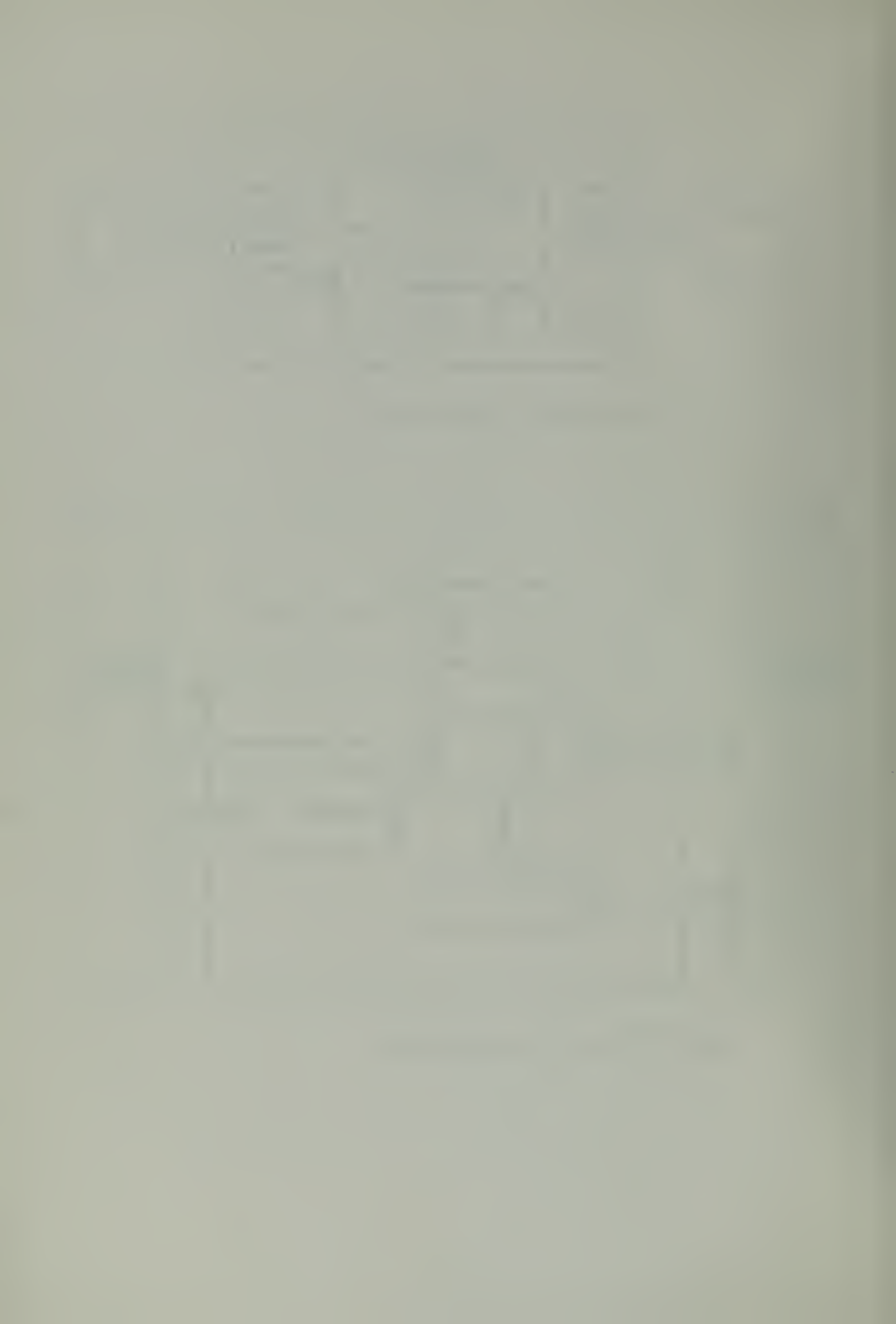


Fig. IV-B-2. Data Processing Unit



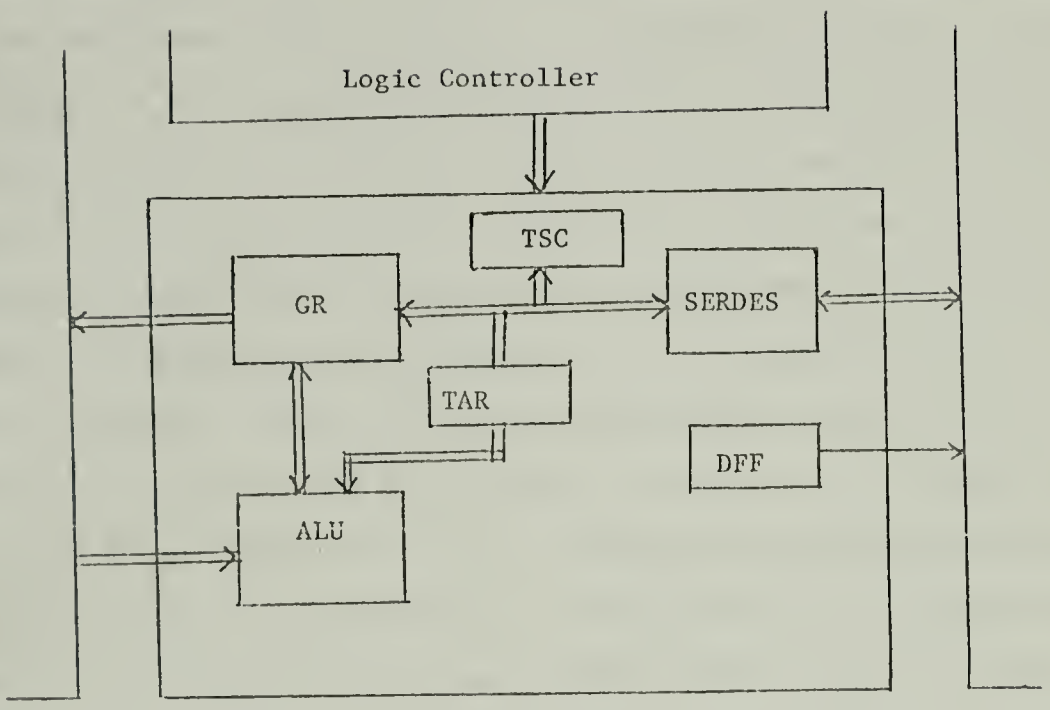
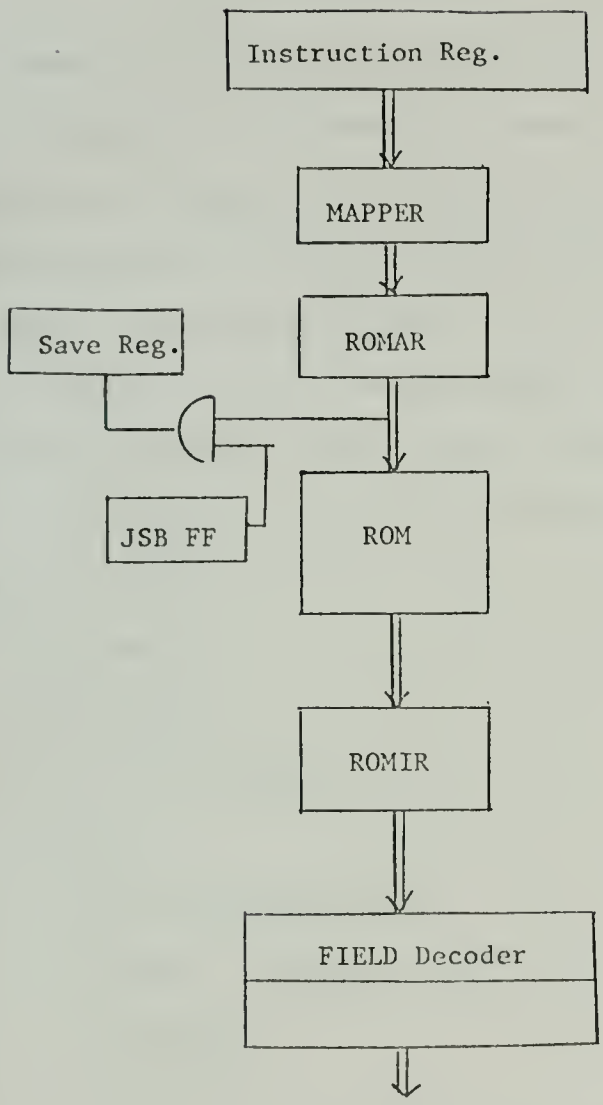


Fig. IV-B-3. Seek Data Flow

2. Microprogrammed Control Unit

The block diagram of the microprogrammed control unit is the same as the conventional one, the basic difference being only in the structure of the controller. The controller logic is now replaced by a microprogrammed controller. The microprogrammed controller (Fig. IV-B-4) consists of read only memory (ROM), where each word describes an action or sets up activities to be performed with the control unit. The configuration of bits in the ROM word describes which gates are to be used, what position the gate is in and hence the flow of information within the control unit (Ref. 4). Control words are brought into the ROM instruction register (ROMIR), and these words are decoded by the field decoder to form control and data signals. The ROM address register (ROMAR) points to the word in ROM which will control the next activity. The address is incremented after execution of each instruction, or if a microprogram jump is received (for example, a JSB), the JSB signal sets the JSB flip flop. This inhibits the save register from further copying of ROMAR contents; thus the return address for the subroutine is saved (Ref. 2). The signals that come out from the field decoder control all activites in the control unit.

The microprogrammed controller has advantages compared to the conventional controller. The most important is its flexibility in instruction set. Since the instruction set is microprogrammed in ROM, any changes in instruction mean only changes in the ROM. This does not redesign the unit at all. Microprogramming is a more orderly approach to control design, therefore errors are fewer and easier to correct. Also the cost is less, since it requires fewer components and consists of mainly the relatively inexpensive ROM.



to DPU and other control

Fig. IV-B-4. Microprogrammed Controller

V. DISK CONTROL UNIT FOR PERSONNEL MANAGEMENT

A. PROBLEM IN PERSONNEL FILE MANAGEMENT

A record for a personnel file consists of fixed data and data that changes periodically or non-periodically. In addition, the length of the data items can be fixed or variable length, depending upon the type of the data. Each person has some data that are constant for him, such as his birthday and social security number (serial number) and some data that can change: number in family, rank, education, etc.

Let us take an example of a personnel file which has several data elements as follows:

1. Name
2. Birthday
3. Serial number
4. Rank
5. Year entering service
6. Family: wife and children
7. Education
8. Duty, etc.

For the purpose of a computer file, we must make a decision as to which data can be implemented with fixed length and which cannot. Obviously elements 1 to 6 above can be fixed length data, while elements 7 and 8 cannot. Also, elements 1, 2 and 3 are fixed data. The above elements can be grouped as follows:

1. Name as field or data item number 1
2. Birthday as field or data item number 2

3. Serial number as field or data item number 3

4. Data as field or data item number 4

Where the first item in the data is rank, second is year entering service. Fields number 1, 2 and 3 can be considered as fixed length fields. Thus, the logical record may look like:

Field name	Bytes required
Name	30
Birthday	6
Serial number	6
Data	350

Thus, for this particular case, each record needs about 400 bytes. If we use a diskette as the storing medium and apply the track format which will be discussed in the next section, each record will occupy four sectors. The best data format that will be used depends upon the kind of file maintenance and uses. Basic file maintenance operations are:

1. Insert new record.

2. Update record.

3. Delete record.

Uses of file might be as follows:

4. Copy data for selected personnel.

5. List names of personnel which have rank of Lieutenant.

The question arises, which are the most frequent, because this will lead to the decision of which data item should have pointers. Let us consider that activities such as number 5 are the most frequent. A list organisation with two pairs of pointers is the most suitable to carry out all

the activities mentioned above. For flexibility in the future the field data will be put on the second, third, and fourth sectors. The track data format for the first sector will contain forward and backward pointers for the record; forward and backward pointers for birthday; name; birthday and serial number (Fig. V-A-1).

DATA AM	BP	FP	BP	FP	NAME	BIRTHDAY	SERIAL NUMBER	

Fig. V-A-1. First Sector Data Format

B. THE DISKETTE STORAGE DRIVES

1. Functional Characteristics

The diskette is a device for recording or storing data, consisting of a flexible disk permanently enclosed in a protective plastic package. This disk is used in a diskette storage drive such as the SA 900 (Ref. 5), which provides storage for 3.1 million bits of data with a data rate of 250 K bits/sec.

The diskette storage device consists of read/write and control electronics, drive mechanism, read/write head, track positioning mechanism, and the removable diskette. These components perform the following functions:

1. Interpret and generate control signals
2. Move read/write head to the selected track
3. Read and write data.

2. Electrical Interface

The interface of the diskette drive can be divided into two categories: signal and power. The signal interface consists of the lines required to control the diskette storage drive and transfer data to and from the unit. There are seven input signal lines to the diskette storage drive.

1. Direction select. This interface defines the direction of motion of the R/W head when the step line is pulsed. The logical one level defines the direction as out, and logical zero level as in.

2. Step. This interface is a control signal which causes the R/W head to move with the direction of motion defined by the direction select line.

3. Load head. This interface line is a control signal to an actuator that allows the disk to be moved into contact with the R/W head.

A logical one level deactivates the head load, and a logical zero activates the head load actuator.

4. File inoperable reset. This interface line provides a direct reset to the "file inoperable" latch. File inoperable is reset with a logical zero level.

5. Write gate. This interface controls the writing of data on the disk. A logical zero on this line enables the write current source and current sink, and disables the stepping circuitry.

6. Write data. This interface line provides the data to be written on the disk.

7. Erase gate. Controls the DC current through the erase element to providestraddle erase which erases the outside fringe of information while writing on the disk. A logical zeroturns on the current and logical one inhibits DC erase current.

There are five output signal lines from the disk storage drive.

1. Track 00. A logical zero signal on this interface line indicates that the R/W head is positioned at track zero.

2. File inoperable is the output of the data safety circuitry.

3. Index. This interface signal is provided by the disk drive once each revolution (166.67 ms) to indicate the beginning of the track.

4. Separated data is the interface line over which data is sent from the diskette.

5. Separated clock provides the using system the clock bits recorded on the disk.

3. Data Format

The basic unit of data in the rest of this report will be a byte. Each byte contains eight binary bits. A single byte can represent one alphameric character (EBDIC) or ASCII).

The logical data format is the data format the system and system user interact with. We will assume for the purpose of this report that the data format for the diskette is comprised of an index track format, data track format, alternate track and spare track. There are 73 data tracks available in the logical format of the diskette. These tracks are numbered 01 to 73. Each track is logically divided into fixed sectors, and the maximum logical length of a sector will be 128 characters.

The physical data format is the format the controller circuitry must interface with. The elements of physical data format are the physical index hole, index mark, sector address mark, sector header and data sectors.

The index mark and sector address mark are recorded with unique clock patterns. The purpose is for their detection by the control unit. One sector of physical data format consists of an identification record, data field and gaps. The identification record consists of an identification address mark, track address, zeros, sector address and cyclic redundancy check (CRC). This CRC and the CRC generated in the SERDES are compared to detect the error.

C. FUNCTIONS

In personnel file management there are some functions to be performed. Such functions are:

1. Insert record
2. Update record
3. Delete record
4. Copy record
5. List name with specified data.

To perform all the above functions the controller has a variety of commands or internal functions. These internal functions are:

1. Read track and sector
2. Seek
3. Search sector
4. Search field
5. Search field low
6. Write field
7. Write record
8. Read field
9. Read record
10. Write data/delete data.

Read track and sector: Read track and sector and store in Track Address Register (TAR) and Sector Address Register (SAR).

Seek: Move the R/W head to the desired track and sector.

Search field: Search the field which is specified by the start byte, length of the field (in bytes), sector number and the field itself. Also store address (track and sector) and pointers (forward and backward) of the current record searched.

Search field low: Search for the next lower field.

Write field: Write the data on the disk with specified starting byte, length of field and sector number.

Write record: Write the whole record.

Read field: Read data from the disk with specified starting byte, length and sector number. Also store the current address and pointers of the field.

Read record: Read the whole record.

D. SYSTEM IMPLEMENTATION

A block diagram of a microprogrammed diskette controller is shown in Fig. V-D-1. A single line indicates a transfer of a bit at a time or a

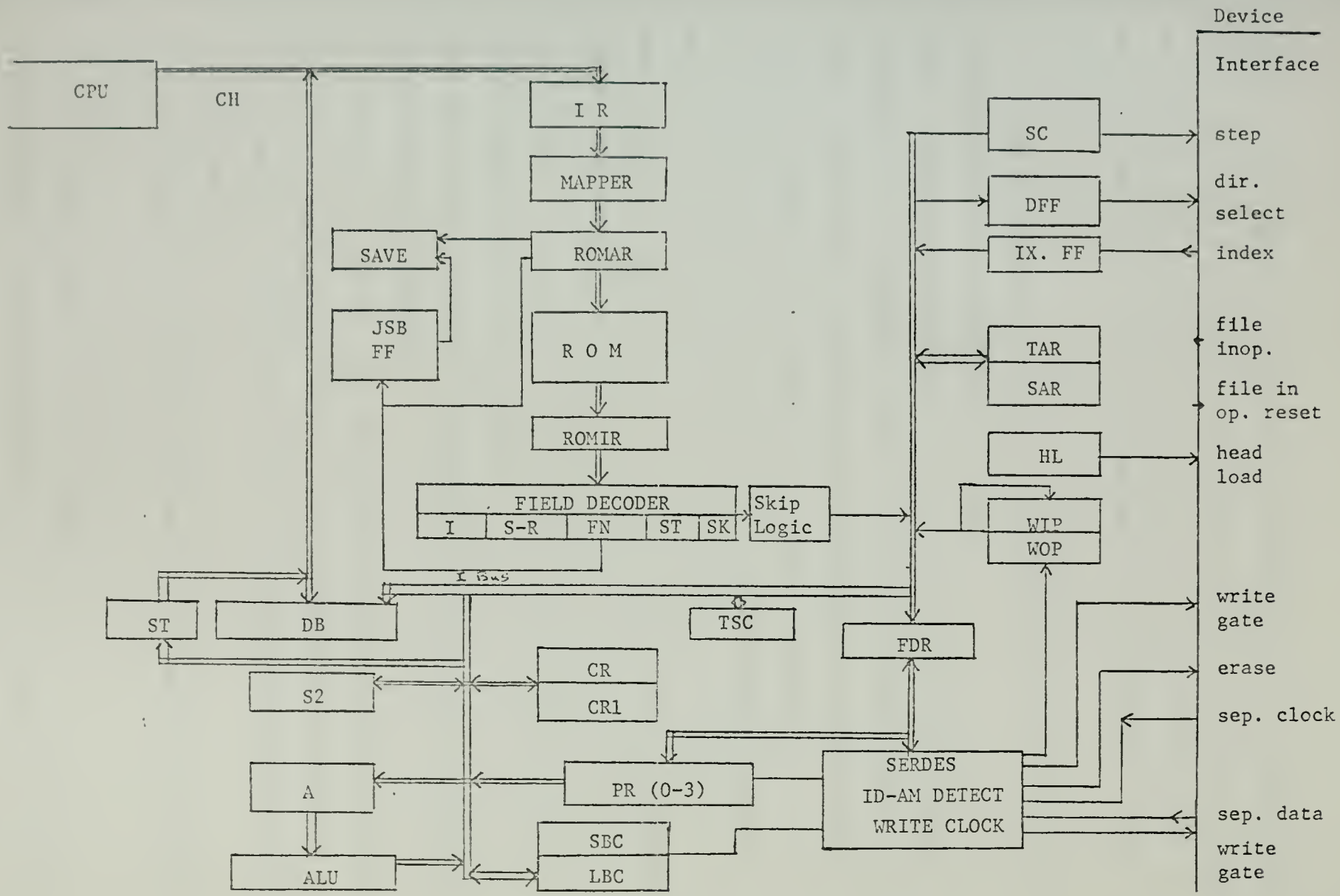


Fig. V-D-1. Microprogrammed Control Unit

pulse, while a double line indicates the transfer of data as several bits in parallel. The data link with the computer is via the eight bit channel bus (CH bus). The path of data and information inside the controller is via the internal bus (I-bus). The components of the controller are as follows:

IR: Instruction Register. Receives instruction from CPU. Holds instruction throughout its execution for decoding.

MAPPER: Decodes the instruction word to form an eight bit ROM address.

ROMAR (ROM Address Register): Holds the addressed microinstruction currently being executed. The address is incremented every clock cycle, or forced to another value according to the current instruction.

ROM (Read Only Memory): Contains microprogram of the basic instruction set.

ROMIR (ROM Instruction Register): Contains the microinstruction from the ROM location addressed by ROMAR.

Field Decoder: Decodes the five fields of the microinstruction in the ROMIR to form control and data signals.

Skip Logic: Logic necessary for testing the skip condition and changing the ROM address accordingly.

Save Register and JSB FF: The save register provides a means for returning from a microprogram subroutine. It is controlled by the JSB FF.

DB (Data Buffer): Intermediate register, for temporarily storing data between CPU and FDR.

ST (Status Register): Contains Ready/Busy bit, Success/Fail bit and ON/OFF bit.

CR, CR1: General purpose eight bit registers.

A (A Register): General purpose register used in conjunction with ALU.

ALU (Arithmatic Logic Unit): Performs arithmetic and logic functions.

S2: A two bit register used to monitor or detect bits, 6 and 7 of a sector address.

PR (Pointer Register): Consists of four 16 bit registers; it receives pointers while reading on each record. Its content can be loaded into the I bus (eight bits each) to indicate track and sector. (FP11, FP12, BP11, BP12, FP21, FP22, BP21, BP22).

SBC, LBC (Starting and Length Byte counters): THeir contents decrease each time the byte comes from SERDES.

TSC (Track Step Counter): Its content decreases every time a pulse arrives from the step circuit.

SC (Step Circuit): Produces necesary pulse forms to move R/W head.

DFF (Direction FF): Content depends on the sign of A.

IX FF: Index FF.

TAR, SAR (Track and Sector Address Registers): Store the track and sector of the current reading.

WID, WOD, WAM (Wait for ID, Zero and AM address/mark FF's): Set by ID, zero and AM detect, reset by controller.

FDR (File Data Register): Temporarily store the data that has been read from the disk.

SERDES, ID-AM DET, WRITE CL: Logic that performs the functions mentioned in the last chapter.

HL (Head Load gate): Controls the head load actuator..

The write gate is inside the SERDES.

The five fields in the field decoder (Appendix A) are as follows: I-bus: controls the internal bus for routing the content of specified registers.

S-R field: Set Reset, cause a pulse to be transferred to the appropriate

register or flip flop. F field: Function field, handles data manipulation and changes the normal addressing. Store field: Stores the content of the I-bus into the specified register or flip flop. Skip field: Causes program to skip next instruction if condition specified is met. A 24 bit word is needed to accommodate the five fields already mentioned, but no work was done in coding the entire orders into specific bit patterns.

The microprograms in the ROM are listed in Appendix B.

An example use of the internal functions for implementing the file operations can be shown as follows: To delete a record, then the program would be as follows:

```
    DELETE RECORD (NAME)
CALL SEARCH FIELD  (SB, LB, S2, NAME)
     CR   BP11
CALL SEEK          (CR)
     SB   0
     LB   2
CALL WRITE FIELD   (SB, LB, BP1)
     CR   FP11
CALL SEEK          (CR)
     CR   FP12
CALL SEARCH SECTOR (CR)
     SB   2
     LB   2
CALL WRITE FIELD   (SB, LB, FP1)
     CR   FP21
CALL SEEK          (CR)
     CR   BP22
```

```
CALL SEARCH SECTOR (CR)
     SB   4
     LB   2
CALL WRITE FIELD   (SB, LB, BP2)
     CR   FP21
CALL SEEK          (CR)
     CR   FP22
CALL SEARCH SECTOR (CR)
     SB   6
     LB   2
CALL WRITE         (SB, LB, FP2)
     CR   T
CALL SEEK          (CR)
     CR   S
CALL SEARCH SECTOR (CR)
     WRITE FIELD DATA
     END
```

To update a record with fixed length:

```
     UPDATE RECORD (NAME)
CALL SEARCH FIELD  (SB, LB, S2, NAME)
CALL WRITE DATA    (SB, LB, S2, CATA)
     END
```

The functions mentioned in Section B of this chapter only deal with fixed-length data. The controller unit can also be used for updating variable-length data. In this case a large eight bit parallel shift register is needed. This shift register is for temporarily storing the data which is located after the data which will be updated. After the updated data

is written on the disk the temporarily stored data can be written back to the disk. With this capability the control unit is not only appropriate for personnel file management but also for such problems as inventory of equipment in a warehouse or store.

VI. CONCLUSIONS

In a conventional controller, all activities are carried out by main computer memory, CPU, control unit, channel and the device drive (disk drive). In the case of personnel file management operation, there are large amounts of data shuffling operations; i.e, transferring data from the disk to the main memory, modifying and transferring the data back to the disk again. This repetitive work takes a lot of CPU time. If the microprogrammed controller is used instead of a conventional one, this repetitive work is carried out by the controller. This can save the CPU from doing the busy repetitive work, and make that time available for other operations.

In a conventional controller, the cost of the total system depends on the cost of the basic logic blocks, on the complexity of the cycle, and the number of cycles. Increasing the machine capability by increasing the complexity of the system will increase the total amount of hardware, thus increasing the cost. This increasing of the cost is proportional to the increase of the machine's capability.

The cost of a microprogrammed controller is determined by the cost of the ROM array, address register and decoder, ROM instruction register, field decoder and sense amplifiers. Increasing the machine capability does not mean increasing all the above components. The only possible change would be in the ROM address register (need more bits) and the associated decoding system.

This control unit does not have to be attached to the large central computer; it can be attached to a mini- or microcomputer. Thus the system will be cheap, easy to expand, and flexible.

In this report, principles of microprogramming and data management systems were presented. Next the conventional and microprogrammed control units were presented and compared. Personnel file management is an example of a computer file management activity which requires a large amount of data shuffling operations. The simplified design of a microprogrammed diskette controller and a microprogram for personnel file operation were developed. The programs for implementation of some of the personnel file maintenance activities demonstrate the flexibility of a microprogrammed system. These and other advantages of the microprogrammed controller lead to the conclusion that microprogrammed controller applied to a direct access peripheral device for data base management would be an effective solution to some processing jobs.

APPENDIX A

MICROPROGRAM FIELDS

I - BUS

TAR track address register

SAR Sector address register

A A register

TSC track step counter

DB data buffer

FDR file data register

SBC start byte counter

LBC length byte counter

WID wait for ID FF

WOD wait for zero FF

WAM wait for AM FF

CR constant register

CR1 constant register

CH channel

CPU CPU data register

SR sign bit of A Reg. (bit 7)

S2 bits 6 and 7 register

SARZ bits 6 and 7 of SAR

BP1 bits 0-7 of backward pointer (BP) register

BP2 bits 8-15 of BP

FP1 bits 0-7 of forward pointer (FP) register

FP2 bits 8-15 of FP

0 all zeros on I bus

1 all ones on I bus

S - R FIELD

SBC set start byte counter

LBC set length byte counter

HL head load control gate

WG write gate control

WID reset WID FF

WOD reset WOD FF

WAM reset WAM FF

PR set PR

SC set step counter

FUNCTION FIELD

SUB content of I bus subtracted from A Reg. and store the result in A Reg.

JMP Uses the 'STORE' and 'SKIP' field contents as the next value for ROMAR

JSB same as JMP, with return address transferred to save register

RPT next instruction to be repeated until conditions in skip field are met

RSB next instruction address is that in save register

STORE FIELD

A A reg.

TSC track step counter

TAR track address reg.

SAR sector address reg.

S2 bits 6 and 7 reg.

DF direction FF

CR constant register

ST1 status register bit 1

DB data buffer

SBC start byte counter

LBC length byte counter

SKIP FIELD

AZR skip next instruction if A=0

NEG skip next instruction if A 0

ODD skip next instruction if A=1

UNC skip next instruction unconditionally

EOP reset busy bit in Status Register

APPENDIX B: EXAMPLE MICROPROGRAMS

READ TRACK AND SECTOR

Address		I-BUS	S-R	FUNC.	STORE	SKIP	COMMENTS
RDTS	0100	NOP	HL	RPT	NOP	NOP	activate HL, set repeat mode.
	0101	WID	WID	NOP	A	ODD	reset WID FF, wait until set by ID detect.
	0102	FDR	NOP	RPT	TAR	NOP	store FDT into Ibus and then to TAR, set repeat mode.
	0103	WOD	WOD	NOP	A	ODD	reset WOD FF, wait until set by WO detect
	0104	FDT	NOP	NOP	SAR	NOP	store FDR into I bus and then to SAR, set repeat mode.
	0105	NOP	HL	RSB	NOP	NOP	deactivate HL, return from subroutine.

SEEK

SEEK	0110	CR	NOP	NOP	A	NOP	store CR into A Reg.
	0111	NOP	NOP	JSB	(RDTS)		jump to subroutine RDTS
	0112	TARE	NOP	SUB	A	NOP	subtract TAR from A and store the result in A.
	0113	SR	NOP	NOP	DF	NOP	store sign of A(SR), into DF.
	0114	A	NOP	RPT	TSC	NOP	store A into TSC and set repeat mode.
	0115		SC	NOP	A	AZR	store TSC into A, set SC, skip if A is zero.
	0116	NOP	NOP	RSB	NOP	NOP	return from subroutine.

Address		I-BUS	S-R	FUNC.	STORE	SKIP	Comments
SEARCH SECTOR							
SESE	0117	NOP	NOP	JSB (RDTS)			jump to subroutine RDTS.
	0118	CR	NOP	NOP	A	NOP	store CR into I bus and then store into A.
	0119	SAR	NOP	SUB	A	AZR	subtract SAR from A, store in A, skip next step if A is zero.
	0120	NOP	NOP	JMP (SESE)			jump to SESE.
	0121	NOP	NOP	RSB	NOP	NOP	return from subroutine.
READ FIELD							
RDF	0122	CR	NOP	NOP	NOP	SBC	store CR into SBC.
	0123	CR	NOP	NOP	NOP	LBC	store CR1 into LBC.
	0124	NOP	NOP	JSB (RDTS)			jump to subroutine RDTS.
	0125	NOP	HL	RPT	NOP	NOP	activate HL, set repeat mode.
	0126	WAM	NOP	RPT	A	ODD	reset WAM FF, wait until set by WAM detect, set RPT.
	0127	SBC	SBC,PR	NOP	A	AZR	set SBC and store in A, set PR, skip next step if A is zero.
LRDF	0128	LBC	LBC	NOP	A	AZR	set LBC and store in A, set skip next step if if A is zero.
	0129	FDR	NOP	NOP	DB	UNC	store FDR in I bus then to DB, skip next step uncond.
	0130	NOP	HL	RSB	NOP	NOP	deactivate HL, return from subroutine.

READ FIELD cont.

Address		I-BUS	S-R	FUNC.	STORE	SKIP	Comments
LRDF	0131	DB,CH	NOP	NOP	NOP	CPU	store DB into CH, and then store in CPU.
	0132	NOP	NOP	JMP	(LDRF)		jump to LRDF.

SEARCH FIELD

SDF	0135	'CPU,CH'	NOP	NOP	DB	NOP	store CPU into CH and then into DB.
	0136	DB	NOP	NOP	SBC	NOP	store DB into SBC.
LSF1	0137	'CPU,CH'	NOP	NOP	DB	NPO	store CPU into CH and then into DB.
	0138	DB	NOP	NOP	LBC	NOP	store DB into LBC.
	0139	'CPU,CH'	NOP	NOP	DB	NOP	store CPU into CH and then into DB.
	0140	DB	NOP	NOP	S2	NOP	store DB into S2.
LSF2	0141	NOP	NOP	JSB	(RDTS)		jump to subroutine RDTS.
LSF3	0142	S2	NOP	NOP	A	NOP	store S2 into A.
	0143	'SAR2	NOP	SUB	A	AZR	store SAR2 into I bus and subtract from A.
	0144	NOP	NOP	JMP	(LSF2)		jump to LSF2.
	0145	NOP	HL	RPT	NOP	NOP	activate HL; set repeat mode.
	0146	WAM	WAM	RPT	A	ODD	reset WAM and wait until set by AM detect.
	0147	SBC	SBC,PR'	NOP	A	AZR	set SBC and store in A, set PR.

SEARCH FIELD cont.

Address		I-BUS	S-R	FUNC.	STORE	SKIP	Comments
LSF4	0148	LBC	LBC	NOP	A	AZR	set LBC and store in A, skip next step if A is zero.
	0149	FDR	NOP	NOP	A	UNC	store FDR into A and skip next step uncond.
	0150	NOP	NOP	JMP (YF)			jump to YF.
	0151	'CPU,CH'	NOP	NOP	DB	NOP	store CPU into CH and then into DB.
	0152	DB	NOP	SUB	A	AZR	store DB into I bus and then subtract from A.
	0153	NOP	NOP	JMP (LSF5)			jump into LSF5.
	0154	NOP	NOP	JUMP (LSF4)			jump into LSF4.
LSF5	0155	0	NOP	SUB	A	NEG	subtract zero from A, skip next step if A is neg.
	0156	NOP	NOP	JMP (LSF6)			jump into LSF6.
	0157	BP1	NOP	NOP	A	NOP	store BP1 into A.
	0158	0	NOP	SUB	A	AZR	subtract zero from A, skip next step if A is zero.
	0159	BP1	NOP	NOP	CR	UNC	store BP1 in CR, skip next step uncond.
	0160	NOP	NOP	JMP (NF)			jump to NF.
	0161	NOP	NOP	JSB (SEEK)			jump to subroutine SEEK.
	0161	BP2	NOP	NOP	CR	NOP	store BP2 into CR.
	0162	NOP	NOP	JSB (SESE)			jump to subroutine Search Sector.
	0163	NOP	NOP	JMP (LSF3)			jump to LSF3.

SEARCH FIELD cont.

Address		I-BUS	S-R	FUNC.	STORE	SKIP	Comments
LSF6	0164	FP1	NOP	NOP	A	NOP	store FP1 into A.
	0165	0	NOP	SUB	A	AZR	subtract zero from A, skip next step if A is zero.
	0166	FP1	NOP	NOP	CR	UNC	store FP1 into CR, skip next step uncond.
	0167	NOP	NOP	JMP	(NF)		jump to NF.
	0168	NOP	NOP	JSB	(SEEK)		jump to subroutine SEEK.
	0169	FP2	NOP	NOP	CR	NOP	store FP2 into CR.
	0170	NOP	NOP	JSB	(SESE)		jump to subroutine SESE.
	0171	NOP	NOP	JMP	(LSF3)		jump to LSF3.
NF	0172	0	NOP	NOP	ST1	UNC	store zero into status reg. (no field).
YF	0173	1	NOP	NOP	ST1	NOP	store one in ST (yes field).
	0174	NOP	HL	NOP	NOP	EOP	deactivate HL, EOP.

LIST OF REFERENCES

1. Husson, Samir S., Microprogramming Principles and Practices, Prentice Hall, Inc., Englewood Cliff, N.J., 1970.

2. Hewlett Packard Co., Microprogramming Guide for Model 2100 Computer, November 1971.

3. George G. Dodd, "Elements of Data Management Systems," Computing Surveys, Vol. I, No. 2, July 1969.

4. Ivan Flores, Peripheral Devices, Prentice Hall, Inc., Englewood Cliff, N.J., 1973.

5. Shugart Associates, SA 900/901 Diskette Storage Drive.

6. Billing, H., and Hopmann, W., "Mikropgoramm Steuerwerk," Elektronische Rundshau, V9, 1955, p. 349.

7. Glantz, H.T., "A Note on Microprogramming," JACM, 3, 1956, p. 77.

8. Mercer, R.J., "Microprogramming," JACM, 4, 1957, p. 157.

9. Dineen, G.P., and others, "The Logical Design of CG24," Proc. EJCC, December 1958, p. 91.

10. Kampe, T.W., "The design of a General Purpose Microprogram Controlled Computer with Elementary Structure," IRE Trans. EC-9, 1960, p. 208.

11. Gerace, G.B., "Microprogram Control for Computing System," IRE Trans. EC-12, 1963, p. 733.

12. Hagiware, H. and others, "The KT Pilot Computer-A Microprogrammed Computer with a Photo Transistor Fixed Memory," Proc. IFIP Congress 62, 1963, p. 684.

13. van der Poel, W.L., "Zebra, a Simple Binary Computer, IPPICIP, Unesco, 1959 London: Butterworth, 1960, p. 361.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 52 Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
4.	Asst. Prof. V. M. Powers, Code 52Pw Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
5.	MAJ Darmawan USDLG Defense Liaison Group Djakarta APO San Francisco 93956	1